
LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

ArrayLists

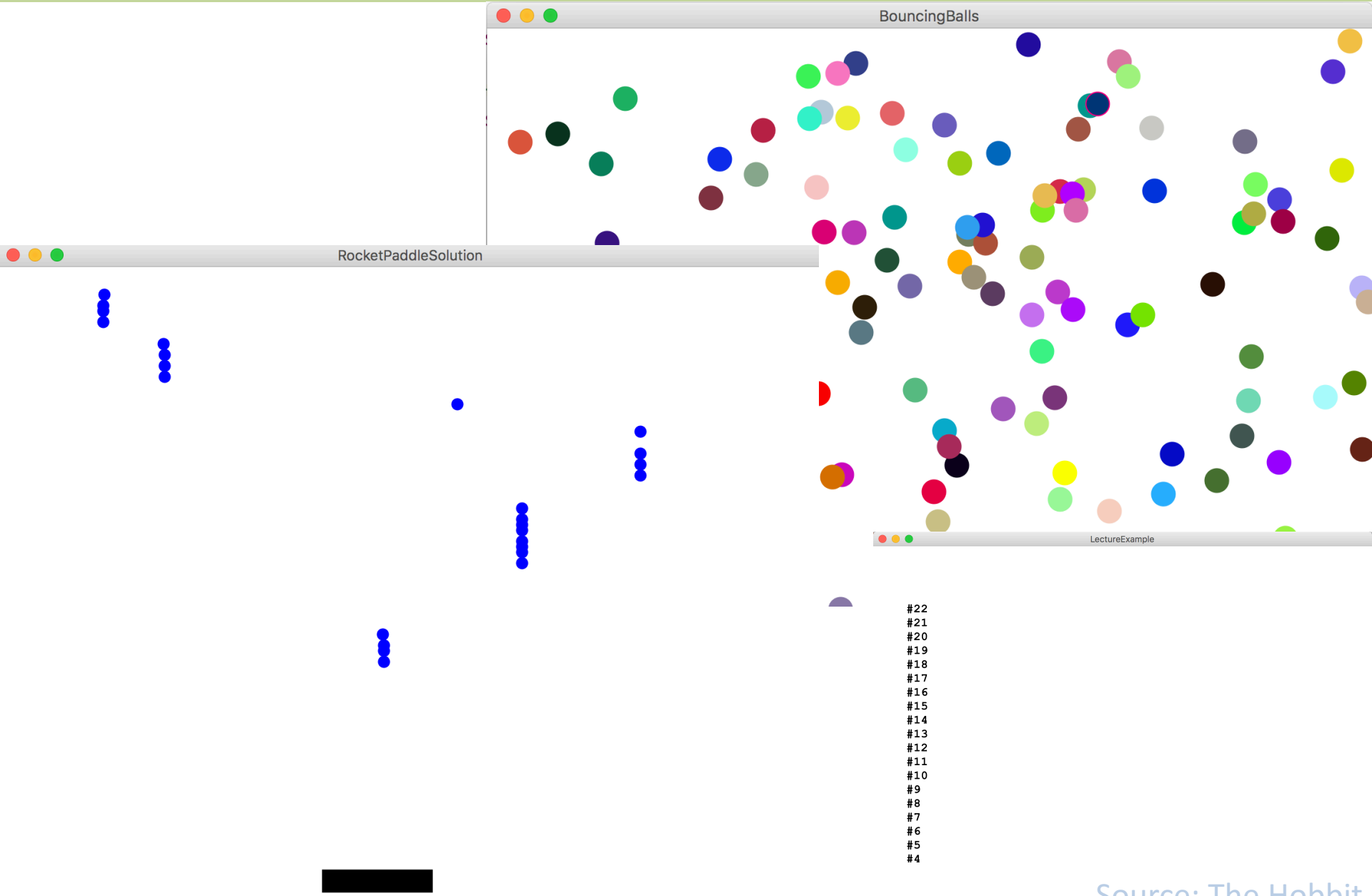
Previously…

- An **array** is a variable type that represents a list of items.
- You access individual items in an array by *index*.
- Store a single type of item (int, double, GRect, etc.)

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	12	49	-2	26	5	17	-6	84	72	3

Source: The Hobbit

After this lecture!



Source: The Hobbit

Learning Goals

1. Know how to store data in and retrieve data from an ArrayList

Meet ArrayLists

- A variable type that represents a list of items.
- You access individual items by index.
- Store a single type of object (String, GRect, etc.)
- **Resizable** – can add and remove elements
- Has helpful methods for searching for items

Wow! Nice to meet you!

ArrayList

```
// Create an (initially empty) list
ArrayList <Integer> list = new ArrayList<Integer>();

// Add an element to the back
list.add(16);  // now size 1
list.add(42); // now size 2
```

ArrayList

```
// Create an (initially empty) list
ArrayList <Integer> list = new ArrayList<Integer>();

// Add an element to the back
list.add(16);  // now size 1
list.add(42); // now size 2

// Access elements by index (starting at 0!)
println(list.get(0)); // prints 16
println(list.get(1)); // prints 42
```

ArrayList

```
// Access elements by index (starting at 0!)
for (int i = 0; i < list.size(); i++) {
    println(list.get(i));
}
```

ArrayList Methods

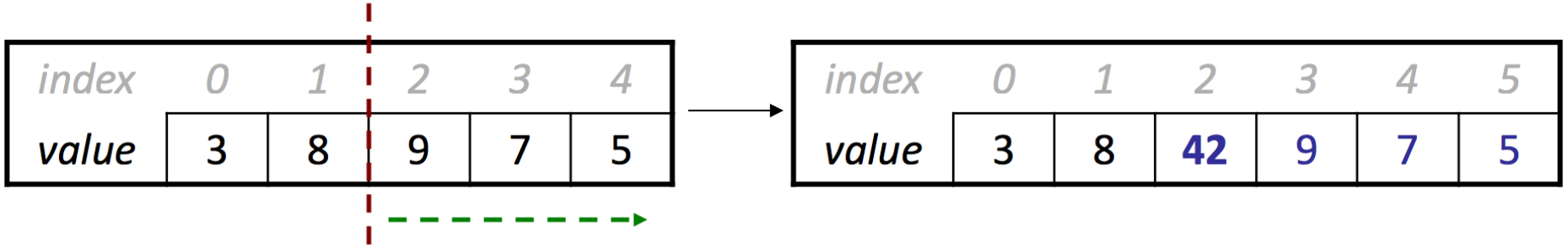
`list.add(value);`	appends value at end of list
`list.add(index, value);`	inserts given value just before the given index, shifting subsequent values to the right
`list.clear();`	removes all elements of the list
`list.get(index)`	returns the value at given index
`list.indexOf(value)`	returns first index where given value is found in list (-1 if not found)
`list.isEmpty()`	returns `true` if the list contains no elements
`list.remove(index);`	removes/returns value at given index, shifting subsequent values to the left
`list.remove(value);`	removes the first occurrence of the value, if any
`list.set(index, value);`	replaces value at given index with given value
`list.size()`	returns the number of elements in the list
`list.toString()`	returns a string representation of the list such as `"[3, 42, -7, 15]"`

Insert/Remove

- If you insert/remove in the front or middle of a list, elements **shift** to fit.

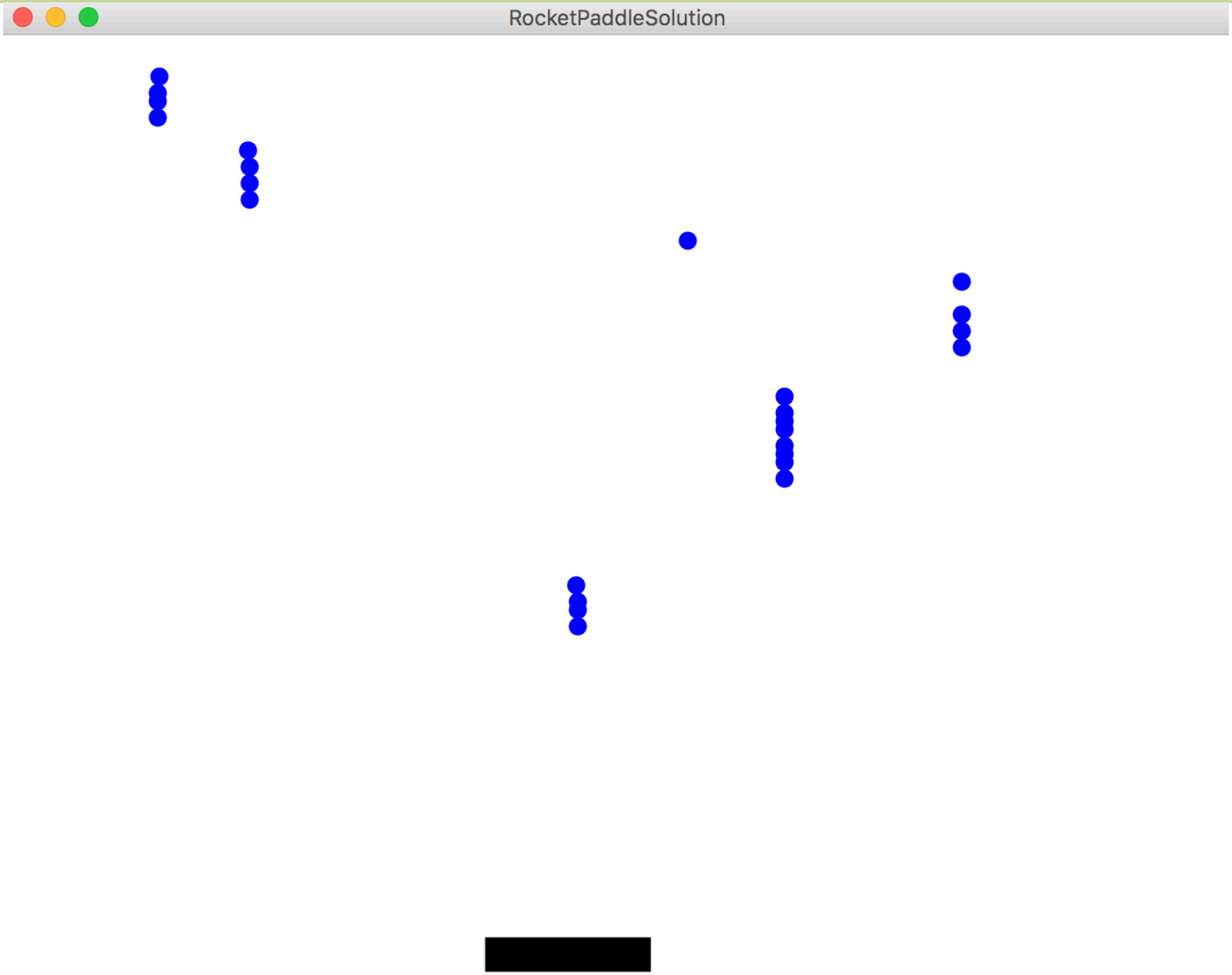
```
list.add(2, 42);
```

  - shift elements right to make room for the new element



Example

Rocket Paddle



ArrayLists and Primitives

```
// Doesn't compile ☹
ArrayList<int> list = new ArrayList<int>();
```

Unlike arrays, ArrayLists can only store **objects!**

ArrayLists and Primitives

Primitive	"Wrapper" Class
`int`	`Integer`
`double`	`Double`
`boolean`	`Boolean`
`char`	`Character`

ArrayLists and Primitives

```
// Use wrapper classes when making an ArrayList
ArrayList<Integer> list = new ArrayList<Integer>();

// Java converts Integer <-> int automatically!
int num = 123;
list.add(num);

int first = list.get(0);    // 123
```

ArrayLists vs. Arrays

Why do both of these exist in the language?

- Arrays are Java's fundamental data storage
- `ArrayList` is a library built on top of an array

When would you choose an array over an `ArrayList`?

- When you need a fixed size that you know ahead of time
  - Simpler syntax for getting/setting
  - More efficient
- *Multi-dimensional* arrays (e.g. images)